**Probleme propuse pentru examenul de ATESTAT la INFORMATICĂ**

**Secțiunea PROGRAMARE**

1. Din fişierul text *vector.in*  se citesc: un număr natural **n** reprezentând numărul de componente ale unui vector cu numere naturale şi cele **n** componente ale vectorului (n<50000). **Fiecare componentă va fi un număr natural cu cel puţin două cifre!**
2. Afişaţi elementele vectorului pe primul rând al fişierului text *vector.out*, separate prin câte un spaţiu.
3. Sortaţi crescător vectorul, după ultimele două cifre ale fiecăruia din cele **n** numere ce reprezintă componentele vectorului, apoi afişaţi, pe al doilea rând al fişierului *vector.out* componentele vectorului sortat, separate prin câte un spaţiu.

**Exemplu:** Dacă fişierul *vector.in* are forma:

5

122 13 1445 9000 321

atunci fişierul *vector.out* va conţine:

122 13 1445 9000 321

9000 13 321 122 1445

1. Din fişierul text *vector.in*  se citesc: un număr natural **n** (n<10000) reprezentând numărul de componente ale unui vector cu numere naturale şi cele **n** componente ale vectorului.
2. Afişaţi componentele vectorului, pe primul rând al fişierului text *vector.out*, în ordine inversă citirii lor .
3. Pe al doilea rând al fişierului text *vector.out*afişaţi media aritmetică a componentelor vectorului, care sunt divizibile cu 3. Există cel puțin un număr multiplu de 3.

**Exemplu**: Dacă fişierul *vector.in*are forma:

5

12 6 48 28 33

atunci fişierul *vector.out* va conţine:

33 28 48 6 12

24.75

1. Din fişierul text *vector.in*  se citesc: un număr natural **n (n<2000)** şi **n** perechi de numere **a**, **b** sunt două numere întregi ce reprezintă capetele unor intervale închise la ambele capete, de forma [**a,b**] (a<b).

Pe primul rând al fişierului text *vector.out*, afişaţi suma obţinută prin adunarea tuturor numerelor întregi aflate în intervalul obţinut prin intersectarea celor **n** intervale date. Dacă intersecţia este mulţimea vidă, atunci se va afişa mesajul „Niciun element!”.

**Exemplu:** Dacă fişierul *vector.in*are forma:

3

-3 2

-2 10

-2 7

atunci fişierul *vector.out* va conţine:

0

Explicaţie: Intervalul obţinut prin intersectarea celor **n** intervale este [-2,2].

1. Este sfârşitul şcolii. Dirigintele clasei doreşte să premieze elevii cei mai buni. Pentru aceasta are nevoie de o ierarhizare a acestora, în ordinea descrescătoare a mediilor. Ştiind că, pentru fiecare din cei **n** elevi ai clasei se cunosc numele şi media generală, ajutaţi-l pe diriginte să facă premierea.

Datele se citesc din fişierul *vector.in* astfel:

* Pe prima linie se află numărul **n** de elevi;
* Pe următoarele **n** linii se află numele elevului şi media, separate prin câte un spaţiu.

În fişierul de ieşire *vector.out* se vor afişa pe **n** linii, separate prin câte un spaţiu, numele elevului şi media, în ordinea descrescătoare a mediilor.

**Exemplu:** Dacă fişierul *vector.in* are forma:

3

Popoviciu 7.86

Ionovici 9.47

Arhivescu 8. 88

atunci fişierul *vector.out* va conţine:

Ionovici 9.47

Arhivescu 8.88

Popoviciu 7.86

1. Din fişierul *vector.in* se citesc: **n** un număr natural nenul reprezentând numărul de componente ale unui vector cu numere naturale nenule şi cele **n** componente ale vectorului.
2. Afişaţi, pe primul rând al fişierului *vector.out* mesajul „Exista” sau „Nu exista”, dacă printre componentele vectorului există vreun număr perfect. Un număr este perfect dacă este egal cu suma divizorilor săi.
3. În cazul în care, există cel puţin un număr perfect, afişaţi, pe al doilea rând al fişierului *vector.out* suma tuturor numerelor perfecte din şirul dat. Dacă nu există niciun număr cu proprietatea cerută, pe al doilea rând al fişierului de ieşire se va afişa -1.

**Exemplu:** Dacă fişierul *vector.in* are forma:

6

13 33 6 28 6 16

atunci fişierul *vector.out* va conţine:

Exista

40

Explicaţie: Numărul 6 este perfect deoarece 6=1+2+3

1. Din fișierul text *date.in* se citesc numere întregi. Să se determine cel mai mare divizor propriu al fiecărui număr citit. Afișarea va fi făcută în fișierul *date.out.*

**Exemplu :**

|  |  |
| --- | --- |
| *date.in*  6 8 15 21 | *date.out*  Divizorul maxim al elementului 6 este 3  Divizorul maxim al elementului 8 este 4  Divizorul maxim al elementului 15 este 5  Divizorul maxim al elementului 21 este 7 |

1. Din fisierul *date.in* se citesc de pe prima linie un număr natural *n,*  iar de pe următoarele *n* linii o matrice patratică de dimensiune *nxn*. Să se afișeze suma elementelor de pe fiecare linie. Rezultatul va fi afișat în fisierul *date.out* sub forma:

linie 1 - >suma\_1

linie 2 - >suma\_2

.........

linie n ->suma\_n

**Exemplu :**

|  |  |
| --- | --- |
| *date.in*  4  2 5 3 6  3 5 8 4  1 2 6 3  8 3 5 9 | *date.out*  linie 1 - > 16  linie 2 - > 20  linie 3 - > 12  linie 4 - > 25 |

1. Fie sirului Fibonacci dat prin recurența

F0=0, F1=1,…,Fn=Fn-1 + Fn-2.

Să se calculeze recursiv al n-lea termen Fibonacci.

**Exemplu:** pentru n=5 se va afișa 5, iar pentru n=8 se va afișa 21.

1. Din fișierul *contur.in* se citește n, număr natural (2<=n<=20) și o matrice pătratică de dimensiune nxn. Realizați un program care afișează în fișierul *contur.out* :
2. conturul matricei
3. câte elemente prime se află pe conturul matricei

**Exemplu:** Pentru n=3 și matricea:

1 2 3

4 5 6

7 8 9

Se va afișa:

1 2 3 6 9 8 7 4

3

1. Realizați un program care determină cel mai mare divizor comun al elementelor situate pe fiecare coloană a unei matrici pătratice.

**Exemplu:** pentru n=3 și matricea

4 12 36

3 15 90

7 6 30

se va afișa 4 , 3 , 1.

1. Se citește un șir de n (număr natural 1<n<=10) numere naturale. Se formează un număr cu ultima cifră a fiecărui număr din șir. Să se scrie un program care afișează pe ecran dacă numărul astfel obținut este palindrom (adică dacă citit de la stânga la dreapta este egal cu cel citit de la dreapta la stânga).

**Exemplu:** Pentru n=6 și numerele 11 22 43 151 17 3, se va afișa mesajul ”nu este palindrom”.

1. Să se citească din fişiereul text „*date.in*” un şir de maximum 255 de caractere format din cuvinte separate prin unul sau mai multe spaţii. Cuvintele sunt formate numai din litere mici ale alfabetului englez. Scrieţi un program C/C++ care citeşte un astfel de şir şi afişează pe ecran frecvenţa de apariţie a fiecăreu litere din şir.

**Exemplu**:

Pentru şirul: competente profesionale

Se va afişa:

a apare de 1 ori

c apare de 1 ori

e apare de 5 ori

f apare de 1 ori

i apare de 1 ori

l apare de 1 ori

m apare de 1 ori

n apare de 2 ori

o apare de 3 ori

p apare de 2 ori

r apare de 1 ori

s apare de 1 ori

t apare de 2 ori

1. Să se citească de la tastatură două şiruri formate din maximum 25 de caractere fiecare Fiecare şir conţine numai litere mici ale alfabetului englez. Scrieţi un program C/C++ care verifică şi afişează printr-un mesaj sugestiv, dacă cele două şiruri sunt sau nu anagrame (să conţină aceleaşi litere, în orice ordine de apariţie).

**Exemplu:**

Pentru şirurile „curat” si „urcat” se afişează “sirurile sunt anagrame”

Pentru şirurile „problema” si „emblema” se afişează “sirurile nu sunt anagrame”

1. Se dă un vector cu n componente numere întregi (datele se citesc de la tastatură, n<=50). Răspundeţi următoarelor cerinţe, utilizând subprograme recursive:
2. Să se calculeze şi să se afişeze suma componentelor pozitive din vector.
3. Să se verifice dacă componentele vectorului sunt ordonate crescător. Se va afişa un mesaj adecvat.

**Exemplu:** Fie n=5 şi vectorul cu componentele (5,-4,3,7,-1). Se va afişa valoarea 15 şi mesajul “Vectorul nu este ordonat”.

1. Se dă un vector cu n componente numere naturale (datele se citesc de la tastatură, n<=50). Să se înlocuiască componentele neprime din vector cu numărul divizorilor acestora. Se vor utiliza subprogramele recursive: **nrd** care va implementa un algoritm de numărare a divizorilor unui număr natural şi **inlocuire** prin care se vor înlocui componentele neprime din vector.

**Exemplu:** Dacă vectorul citit este (5,6,1,0,15), după înlocuire devine (5,4,1,0,4).

1. Se dă un vector cu n componente numere naturale (datele se citesc de la tastatură, n<=50). Să se afişeze componentele vectorului care conţin o cifră dată, nenulă (cifra se citeşte de la tastatură). Se vor utiliza subprogramele recursive:

* **verific** care va implementa un algoritm pentru verificarea existenţei unei cifre într-un număr;
* **afiş** pentru afişarea componentelor unui vector care conţin o cifră dată.

**Exemplu:** Dacă vectorul citit este (15,621,14,502,155) şi c=5 se vor afişa valorile: 15 502 155.

1. Se dă un graf neorientat cu n noduri. Numărul de noduri şi muchiile grafului se citesc din fişierul „graf.txt”. Să se verifice dacă graful este complet şi să se afişeze un mesaj adecvat. Dacă graful nu este complet să se afişeze muchiile care ar trebui adăugate astfel încât graful să devină complet.

**Exemplu:** Dacă fişierul „graf.txt” are următorul conţinut:

5

1 3

1 4

1 5

2 4

3 5

Se va afişa mesajul “Graful nu este complet” după care se vor afişa muchiile lipsă:

1 2

2 3

2 5

3 4

4 5

1. Se dă un graf neorientat cu n noduri. Numărul de noduri şi muchiile grafului se citesc din fişierul „graf.txt”. Să se verifice dacă graful este regulat. În situaţia în care graful nu este regulat să se afişeze nodurile terminale şi numărul de noduri izolate. (Un graf neorientat se numește **regulat** dacă toate nodurile sale au același grad).

**Exemplu:** Dacă fişierul „graf.txt” are următorul conţinut:

5

1 3

1 5

2 4

3 5

Se va afişa mesajul “Graful nu este regulat” după care se vor afişa nodurile terminale: 2, 4 şi numărul de noduri izolate: 0

1. Se dă graful un graf neorientat pentru care datele se citesc din fişierul „graf.txt”. Să se verifice dacă un şir de m numere citite de la tastatură este lanţ în graf şi să se afişeze un mesaj adecvat. Dacă formează lanţ, să se verifice dacă lanţul este elementar.

**Exemplu:** Dacă fişierul „graf.txt” are următorul conţinut:

5

1 3

1 4

1 5

2 4

3 5

Şirul de numere 2 4 1 3, citit de la tastatură, formează lanț elementar.

Şirul de numere 4 2 3 1 5 nu formează lanț.

1. Se citește de la tastatură o serie de n numere întregi. Afișați primul și ultimul număr par din serie. Afișările se vor face cu text semnificativ, câte o informație pe linie. Dacă nu există cel puțin 2 numere pare în serie se va afișa textul ”Nu exista”.

**Exemple:**

1. n = 6, seria : 6 1 2 7 4 9, se va afișa 6 și 4.
2. n=4, seria : 6 1 3 7 5 9, se va afișa ”Nu exista”.
3. În fișierul text ”atestat.in” se găsesc pe prima linie două numere întregi **n** și **k** (1<**k**<**n**<=1000), iar pe următoarea linie n numere întregi separate prin spații. Afișați în fișierul text ”atestat.out” primele **k** minime din seria de **n** numere de pe linia a doua a fișierului de intrare.

**Exemplu:** n=5, k=3, seria este: 3, 2, -3, 6 și -1, se va afișa -3, -1, 2.

1. Se citesc de la tastatură două numere întregi **n** și **m**. Construiți și afișați pe ecran, linie cu linie o matrice **n\*m** construită după regula: a[i][j] = min(i,j). Numerotarea liniilor, respectiv coloanelor matricei începe de la 1.

**Exemplu:** Pentru n=2 și m=3 se va afișa matricea:

1 1 1

1 2 2

1. Fişierul text „graf.in” conţine pe prima linie un număr natural n reprezentând numărul de vârfuri ale unui graf neorientat, iar pe fiecare din următoarele n rânduri câte n valori de 0 şi 1 separate prin spaţii, reprezentând elementele unei linii a matricei de adiacenţă corespunzătoare grafului.
2. Să se scrie o funcție **grad** ceprimeşte ca parametru un număr natural x şi returnează gradul vârfului x.
3. Să se scrie programul C++ care citeşte datele din fişier şi care afişează în fişirul text „graf.out”, pe primul rând, separate prin câte un spaţiu vârfurile terminale ale grafului, sau mesajul „Nu există”, dacă în graf nu sunt vârfuri terminale, folosind apeluri utile ale subprogramului grad.

**Exemplu:** Dacă fişierul ”graf.in” are forma:

5

0 0 1 0 1

0 0 0 1 1

1 0 0 0 0

0 1 0 0 0

1 1 0 0 0

atunci fişierul „graf.out” va conţine numerele 3 şi 4.

1. Fişierul text „muchii.txt” conţine pe prima linie un număr natural n reprezentând numărul de vârfuri ale unui graf neorientat, pe al doilea rând un număr natural m reprezentând numărul de munchii ale unui graf neorientat, iar pe fiecare din următoarele m rânduri câte două numere naturale, separate pritr-un spaţiu, reprezantând extremităţile unei muchii a grafului.
2. Să se scrie o funcție **grad** ce primeşte ca parametru un număr natural x şi returnează gradul vârfului x.
3. Să se scrie programul C++ care citeşte datele din fişier, construieşte matricea de adiacenţă asociată grafului, şi care afişează în fişirul text „grade.txt”, separate prin câte un spaţiu vârfurile de grad maxim ale grafului, folosind apeluri utile ale subprogramului grad.

**Exemplu:** Dacă fişierul ”muchii.txt” are forma:

5

6

1 2

2 3

2 4

3 4

3 5

4 5

atunci fişierul „grade.txt” va conţine numerele 2, 3 şi 4.

1. Fişierul text „adiacenta.in” conţine pe prima linie un număr natural n reprezentând numărul de noduri ale unui graf orientat, iar pe fiecare din următoarele n rânduri câte n valori de 0 şi 1 separate prin spaţii, reprezentând elementele unei linii a matricei de adiacenţă corespunzătoare grafului.
2. Să se scrie o funcție **grad\_intern** ce primeşte ca parametru un număr natural x şi returnează gradul intern al nodului x.

Funcţia **grad\_extern** primeşte ca parametru un număr natural x şi returnează gradul extern al nodului x.

1. Să se scrie programul C++ care citeşte datele din fişier, şi care afişează în fişirul text „noduri.out”, separate prin câte un spaţiu nodurile grafului care au gradul intern egal cu gradul extern, folosind apeluri utile ale subprogramelor **grad\_intern** şi **grad\_extern**.

**Exemplu:** Dacă fişierul ”adiacenta.in” are forma:

5

0 1 1 1 0

0 0 0 0 1

0 1 0 0 0

0 0 0 0 0

1 0 0 0 0

atunci fişierul „noduri.out” va conţine numerele 3 şi 5.

1. Fişierul text „arce.txt” conţine pe prima linie un număr natural n reprezentând numărul de noduri ale unui graf orientat, pe al doilea rând un număr natural m reprezentând numărul de arce ale unui graf orientat, iar pe fiecare din următoarele m rânduri câte două numere naturale separate prin spaţii, reprezentând arcele corespunzătoare unui graf orientat.
2. Să se scrie o funcție *grad\_intern* ce primeşte ca parametru un număr natural x şi returnează gradul intern al nodului x.

Funcţia *grad\_extern* primeşte ca parametru un număr natural x şi returnează gradul extern al nodului x.

1. Să se scrie programul C++ care citeşte datele din fişier, construieşte matricea de adiacenţă asociată grafului, şi care afişează în fişirul text „izolate.txt”, separate prin câte un spaţiu nodurile izolate ale grafului, sau mesajul „Nu există”, dacă în graf nu sunt noduri izolate, folosind apeluri utile ale subprogramelor grad\_intern şi grad\_extern.

**Exemplu:** Dacă fişierul ”arce.txt” are forma:

5

3

1 2

2 1

2 4

atunci fişierul „izolate.txt” va conţine numerele 3 şi 5.

1. Se citesc numere naturale din fişierul text **„numere.txt”**. Să se afişeze numerele care au proprietatea de palindrom (numărul citit de la dreapta la stânga este egal cu numărul citit de la sânga la dreapta) şi suma cifrelor să fie pară. Se vor folosi: un subprogram **palin** care va implementa un algoritm de determinare a proprietăţii de număr palindrom şi un subprogram care să calculeze suma cifrelor unui număr natural.

**Exemplu**: Pentru numerele 12, 12321, 565, 45, 18, 121 se va afişa 565, 121.

1. Se citesc numere naturale din fişierul **”numere.txt”**. Câte dintre aceste numere au toate cifrele cifre impare. Dacă nu există astfel de numere se va afişa un mesaj corespunzător. Se va folosi un subprogram care să verifice cerinţa din enunţ referitoare la paritatea cifrelor.

**Exemplu:** Pentru numerele 135, 57, 279, 791, 56 se va afişa 3, iar pentru numerele 1234, 56,22, 54, 78 se va afisa mesajul “Nu există nici un număr cu proprietatea cerută”.

1. Se citesc n numere întregi din fişierul **“numere.txt”**. Să se afişeze cel mai mare divizor comun dintre suma tuturor numerelor şi produsul celor cuprinse între 2 numere date. In fişier pe prima linie vor fi valorile pentru n si cele 2 numere date, iar pe urmatoarea linie cele n numere.

**Exemplu**: Pentru numerele

5 17 27

12 34 25 6 21 se va afişa cmmdc(98,525)=7.

1. Fie doi vectori a şi b formaţi din n, respectiv m numere întregi nenule (m,n<=50). Scrieţi un program care citeşte, numărul de elemente şi valorile din vectorul a, respectiv b şi afişează valorile din vectorul a care sunt prime cu toate valorile din vectorul b. Două numere întregi sunt prime între ele dacă cel mai mare divizor comun al lor este 1.

**Exemplu:** Fie n=6, a=(77,35,55,15,121,11) şi m=3, b=(18,20,10). Se vor afişa numerele 77,121,11.

1. Se citesc 2 vectori **a** şi **b** cu **n**, respectiv **m** componente întregi. Să se determine dacă cei 2 vectori sunt identici. Dacă nu, să se afişeze poziţia de unde nu mai coincid elementele.

Vectorii sunt identici dacă au acelaşi număr de elemente şi au aceleaşi valori.

**Exemplu:** Pentru n=3 și elementele: 1 2 3 și m=3 și elementele: 1 2 3, se va afișa mesajul ”Cei doi vectori sunt identici”.

1. Scrieţi un program C/C++ care citeşte din fișierul ”date.in” un număr natural n (1≤n≤20), elementele unei matrice cu n linii şi n coloane, numere întregi din intervalul [-100,100] şi afişează pe ecran media aritmetică a elementelor prime ale matricei, care sunt situate deasupra diagonalei principale. Dacă nu există elemente prime situate deasupra diagonalei principale, programul va afişa mesajul „Nu exista!”.

**Exemplu:** pentru n=4 şi matricea de mai jos se afişează valoarea 3.33333

-1 2 -4 5

0 6 3 1

2 4 2 0

3 -5 1 -3

1. Scrieţi un program C/C++ care citeşte din fișierul ”*date.in*” un număr natural n (1≤n≤50), elementele unei matrice cu n linii şi n coloane, numere naturale din intervalul [0,99999] şi afişează pe ecran numărul elementelor palindrom ale matricei, care sunt situate deasupra diagonalei secundare. Dacă nu există elemente palindrom situate deasupra diagonalei secundare, programul va afişa mesajul „Nu exista!”.

Un număr natural este palindrom dacă numărul citit de la stânga la dreapta este același cu numărul citit de la dreapta la stânga.

**Exemplu:** pentru n=4 şi matricea de mai jos se afişează valoarea 4.

112 121 444 56

102 696 123 17

262 413 221 10

393 585 111 33

1. Scrieţi un program C/C++ care citeşte din fișierul ”date.in” un număr natural n (1≤n≤100), un număr natural m (1≤m≤10) și elementele unei matrice cu n linii şi m coloane, cifre binare, fiecare linie a tabloului reprezintă cifrele unui număr în baza 2. Să se afişeze pe linii separate ale ecranuluinumerele care se obțin prin transformarea fiecărei linii din matrice în numărul corespunzător în baza 10.

**Exemplu:** Dacă fișierul conține

3 7

1 1 1 1 0 0 1

1 0 1 0 1 0 1

1 0 0 0 1 1 0

Atunci se va afișa:

121

85

70